Theoretical Computer Science 61 (1988) 175-198
North-Holland

ALGORITHMS FOR PLANE REPRESENTATIONS OF ACYCLIC DIGRAPHS*

Giuseppe DI BATTISTA

Dipartimento di Informatica e Sistemistica, University of Rome "La Sapienza", 00185 Rome, Italy

Roberto TAMASSIA**

Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, U.S.A.

Communicated by G. Ausiello
Received March 1987
Revised July 1987

Abstract. Acyclic digraphs are widely used for representing hierarchical structures. Examples include PERT networks, subroutine-call graphs, family trees, organization charts, Hasse diagrams, and ISA hierarchies in knowledge representation diagrams. We investigate the problem of representing acyclic digraphs in the plane in such a way that all edges flow in the same direction, e.g., from the left to the right or from the bottom to the top. Three plane representations are considered: *straight drawings*, *visibility representations*, and *grid drawings*. We provide efficient algorithms that construct these representations with all edges flowing in the same direction. The time complexity is $O(n)$ for visibility representations and grid drawings, and $O(n \log n)$ for straight drawings, where n is the number of vertices of the digraph. For covering digraphs of lattices, the complexity of constructing straight drawings is $O(n)$. We also show that the planar digraphs that admit any one of these representations are exactly the subgraphs of planar *st-graphs*.

1. Introduction

Graphs are used as modeling tools in several areas and therefore the problem of automatically generating plane representations of graphs has many applications. One of the most natural ways to draw graphs in the plane consists of representing vertices by means of points, and edges by means of straight-line segments. As independently shown by Fary [13], Stein [22], and Wagner [32], every planar graph can be drawn in this way without crossings between edges. This classic result has been followed by a series of investigations on planar drawings of graphs with straight-line segments. Tutte [30] proved that every 3-connected planar graph admits a *convex drawing*, i.e., a planar straight-line drawing such that all the face boundaries

* Work partially supported by Istituto Centrale di Statistica (ISTAT), Progetto Finalizzato Trasporti CNR, and National Science Foundation Grant ECS-84-10902.

** Present affiliation: Department of Computer Science, Brown University, Providence, RI 02912, U.S.A.

are convex polygons. He also gave a method for finding a convex drawing, where each vertex is placed in the barycenter of its neighbors, and the external face is any prescribed convex polygon [31]. A linear-time algorithm for constructing convex drawings was presented by Chiba et al. [4, 6]. This algorithm is based on a characterization given by Thomassen [27] of the class of planar graphs that admit a convex drawing. Further results on planar straight-line drawings are described in [14, 27, 28].

A *grid drawing* of a planar graph G is a drawing of G such that the vertices are placed at grid points, and the edges are drawn as polygonal lines that bend only at grid points. The number of bends along the edges and the area occupied are important quality measures for a grid drawing in applications such as circuit layout and automatic graph drawing [1, 23, 24, 26]. This representation has been investigated in [35], where it is shown that every n-vertex planar graph admits a grid drawing with $O(n^2)$ area and $O(n^2)$ bends.

A *visibility representation* for a planar graph G consists of representing the vertices of G by means of horizontal segments, and the edges of G by means of vertical segments, so that the edge-segment associated with an edge (v, w) has its endpoints on the vertex-segments associated with v and w, and does not intersect any other vertex segment [18, 20, 25]. This representation is very convenient for the display of planar graphs because of its simplicity and regularity.

In this paper, we investigate planar drawings of acyclic digraphs. Acyclic digraphs are widely used to represent hierarchic structures. Examples include PERT networks, subroutine-call graphs, family trees, organization charts, Hasse diagrams, and ISA hierarchies in knowledge representation diagrams. In order to visualize the hierarchic structure of these graphs, it is desirable to draw them in such a way that all the edges follow a common general direction, e.g., from bottom to top, or from left to right. We can formalize this concept by defining a *monotonic drawing* as a planar drawing of a digraph such that all edges are curves monotonically increasing in the vertical direction.

Several plane representations for acyclic digraphs with the above monotonic property can be defined by imposing further constraints on the aforementioned plane representations for undirected planar graphs: When using planar straight-line drawings, we impose that the projection of every edge on the vertical axis is positive, and we call a drawing with such a property an *upward drawing* (see Fig. 1). Also, we extend the definition of grid drawings, and call *monotonic grid drawing* a grid drawing which is also a monotonic drawing (see Fig. 2). Finally, in the case of visibility representations, we add the constraint that every edge-segment is directed from the lower to the higher endpoint, and call this representation of digraphs a *directed visibility representation* (see Fig. 3).

Upward drawings have been investigated in the framework of the theory of posets and lattices (for the terminology see, for example, [2]). Kelly and Rival [15] provided a characterization of *planar lattices*, i.e., lattices whose covering digraph admits an upward drawing, in terms of a family of forbidden subposets. Platt [19] showed that a lattice is planar if and only if the undirected graph obtained from

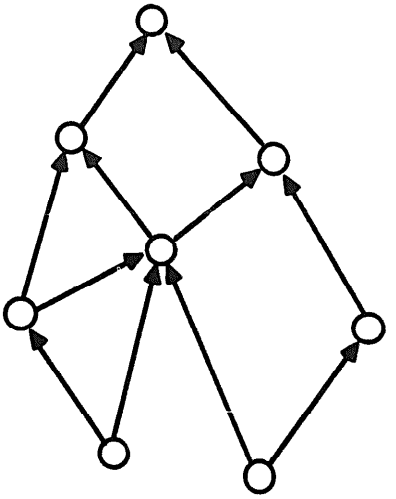

Fig. 1. Example of upward drawing.

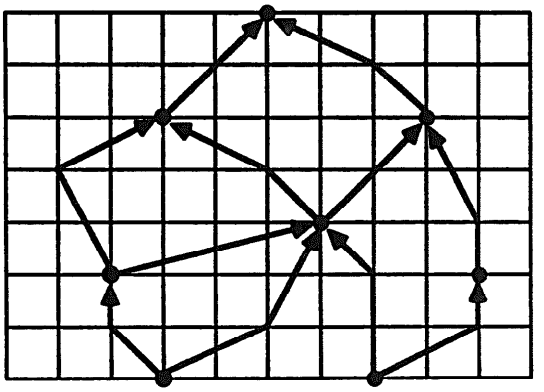

Fig. 2. Example of monotonic grid drawing.

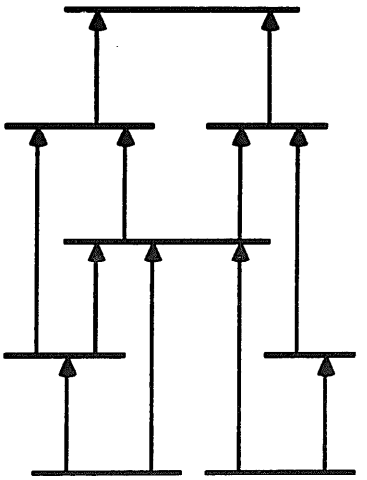

Fig. 3. Example of directed visibility representation.

the covering digraph by ignoring arc directions and adding an edge between its least and greatest elements is planar. The relationship between the dimension of posets and planarity has been investigated in [16, 29].

We provide efficient algorithms for constructing plane representations of acyclic digraphs, whose time complexity is $O(n \log n)$ for upward drawings, and $O(n)$ for monotonic grid drawings and directed visibility representations. We also give an algorithm for constructing the upward drawing of the covering diagram of a planar lattice that runs in $O(n)$ time. Notice that the algorithm for monotonic grid drawings produces a drawing with $O(n^2)$ area and $O(n)$ bends, and is therefore suitable for practical applications.

Finally, we derive a characterization of the class of acyclic digraphs that admit the aforementioned representations. Namely, for any digraph G we show that the following statements are equivalent:

(1) G is a subgraph of a planar *st-graph*;
(2) G admits a monotonic drawing;
(3) G admits a monotonic grid drawing;
(4) G admits an upward drawing;
(5) G admits a directed visibility representation;

where an st-graph is defined as an acyclic digraph with exactly one source, s, and one sink, t, that contains the edge (s, t).

The rest of this paper is organized as follows: Section 2 contains formal definitions of the plane representations considered, and preliminary results. In Section 3, we develop efficient algorithms for constructing upward drawings, directed visibility representations, and monotonic grid drawings. In Section 4, we present the aforementioned characterization. Section 5 contains drawing algorithms for covering digraphs of lattices. Finally, Section 6 addresses directions of further research and open problems in this area.

Note: After the completion of this research work, we became aware that the equivalence of statements (1), (2), and (4) has been independently discovered by Kelly [17]. Our proof of this characterization is simpler and shorter than the one of Kelly.

2. Preliminaries

Unless otherwise specified, we assume graphs to be finite, simple, and connected.

Let G be a planar graph. A *planar straight drawing* for G is a planar embedding of G such that every edge is represented by a straight-line segment. For simplicity we use the term "edge" both for the edges of G and their corresponding segments in the drawing. If G is directed, an *upward drawing* of G is a planar straight drawing of G such that the projection of every edge on the vertical axis is positive, i.e., if

(u, v) is an edge of G, then vertex v is vertically above vertex u. Clearly, if G admits an upward drawing, then G is acyclic.

A *monotonic grid drawing* for a planar digraph G is a planar drawing of G such that every vertex is placed at a grid point, and every edge is drawn as a polygonal line connecting grid points with increasing ordinate. A *directed visibility representation* for G consists of mapping each vertex v of G into an horizontal segment $\sigma(v)$, and each edge (u, v) into a vertical segment $\sigma(u, v)$, so that, for each edge (v, w), the corresponding edge-segment $\sigma(v, w)$ has lower endpoint on $\sigma(v)$, upper endpoint on $\sigma(w)$, and does not intersect any other vertex-segment $\sigma(u)$, $u \neq v,w$.

Let Γ be a directed visibility representation or a monotonic grid drawing. The *height* and *width* of Γ are defined as the height and width of the smallest rectangle (with sides parallel to the coordinate axes) that covers Γ, and will be denoted by $\textsc{Height}(\Gamma)$ and $\textsc{Width}(\Gamma)$ respectively. If Γ is a monotonic grid drawing, we denote $\textsc{Bends}(\Gamma)$ the total number of bends along its edges. For example, the monotonic grid drawing of Fig. 2 has $\textsc{Height}(\Gamma)=7$, $\textsc{Width}(\Gamma)=8$, and $\textsc{Bends}(\Gamma)=8$.

An *st-graph* is a digraph $G=(V, E)$ such that

(1) G is acyclic;

(2) G has exactly one source, s, and exactly one sink, t; and

(3) G contains the edge (s, t).

A *topological ranking* for G is a mapping ρ of the vertices of G into distinct integers such that, for every edge (u, v), $\rho(u)<\rho(v)$. A topological numbering is a special case of topological ranking where the vertices are mapped to the integers $\{1, \ldots, |V|\}$, and can be computed in time $O(|V|+|E|)$ using standard graph search techniques.

Let G be a planar st-graph embedded in the plane so that the edge (s, t) appears on the external face. The following lemmas are proved in [25].

Lemma 2.1. *Let v be a vertex of G. The outgoing (ingoing) edges incident upon v appear consecutively around v (see Fig. 4(a)).*

Lemma 2.2. *Let f be a face of G. The boundary of G consists of two directed paths. Also, given any topological ranking for G, the origin and destination of these two paths are the vertices of the boundary of f with lowest and highest rank respectively (see Fig. 4(b)).*

We call the *dual* of G the digraph G^* defined as follows (see Fig. 5):

(1) vertices of G^* are the faces of G, where we denote s^* and t^* the faces to the right and left of edge (s, t) respectively;

(2) there is an edge (f, g) in G^* if face f shares an edge $(v, w) \neq (s, t)$ with face g, and face f is on the left side of (v, w), when (v, w) is traversed from v to w;

(3) finally, G^* contains the edge (s^*, t^*).

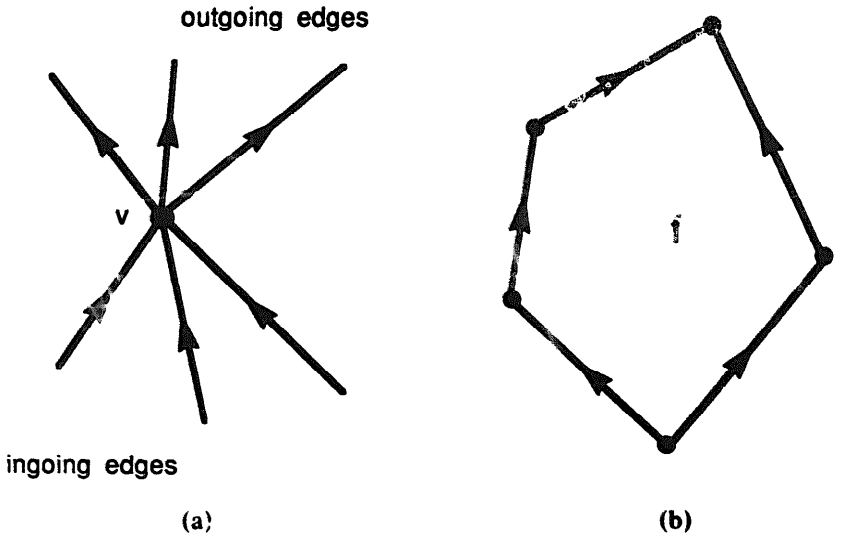

Fig. 4. (a) Example for Lemma 2.1. (b) Example for Lemma 2.2.

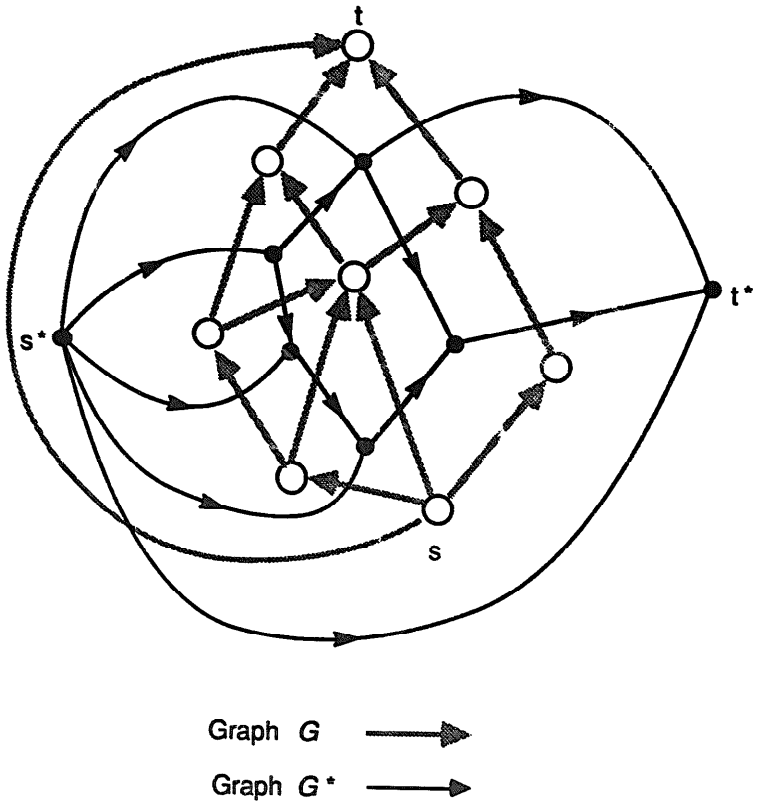

Fig. 5. Example of the dual of an st-graph.

Notice that G^* is an orientation of the dual of the undirected version of G.

Lemma 2.3. *The dual graph of a planar st-graph is a planar st-graph.*

We conclude this section with some geometric definitions that will be used in subsequent constructions.

Let $P=(v_0,\ldots,v_k)$ be a simple polygon, where the sequence of vertices is counterclockwise, and p a point inside P. A vertex v_i of P is said to be *visible* from p if the segment $p-v_i$ lies entirely inside P. A vertex v_i is said to be *properly visible* from p if it is visible from p and the interior of the segment $p-v_i$ does not intersect P. The *kernel* of P is the set of points p from which all vertices of P are visible. Notice that the concept of visibility used here (visibility in all directions) is different from the one of the definition of directed visibility representation (visibility in the vertical direction).

The *left half-plane* of an edge (v_i, v_{i+1}) of P is defined as the half plane on the left of the straight line through (v_i, v_{i+1}), oriented from v_i to v_{i+1}. The *wedge* of a vertex v_i of P is defined as the intersection of the left half-planes of the edges incident upon v_i (see Fig. 6). It has been shown that the kernel of P is equal to the intersection of the left half-planes of its edges, or, equivalently, to the intersection of the wedges of its vertices [36]. Using this characterization of the kernel and continuity arguments we can prove the following lemma (see Fig. 7).

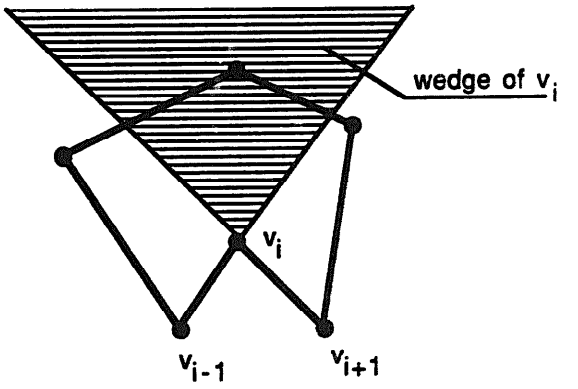

Fig. 6. Wedge of vertex v_i.

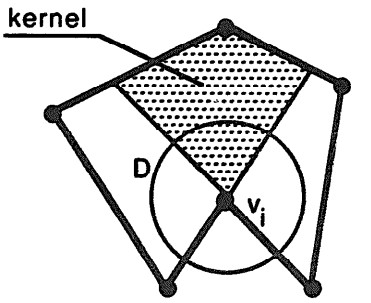

Fig. 7. Example for Lemma 2.4.

Lemma 2.4. *Let v_i be a vertex of polygon P such that*

(1) *v_i belongs to the kernel of P;*

(2) *the remaining vertices of P, except v_{i-1} and v_{i+1}, are properly visible from v_i.*

Then there exists a disk D of radius $\varepsilon > 0$ centered at v_i such that the intersection of D and the wedge of v_i is contained in the kernel of P.

3. Drawing algorithms

In this section, we consider the problem of constructing plane representations of planar acyclic digraphs. We show that, given an n-vertex planar st-graph G, we can compute an upward drawing for G in time $O(n \log n)$, and a directed visibility representation or a monotonic grid drawing for G in $O(n)$ time.

A *triangular st-graph* is a maximal planar st-graph. Notice that in any planar embedding of a triangular st-graph, the boundary of each face consists of exactly three edges. First, we will show that every triangular st-graph admits an upward drawing where the external face is a prescribed triangle by modifying the construction of Fary [13]. To describe our proof, we need further terminology. Given an upward drawing Γ, we denote by $\theta(e)$ the slope of edge e of Γ with respect to the x-axis. The definition of upward drawing is equivalent to saying that $0 < \theta(e) < \pi$ for all edges e of Γ. Let θ_1 and θ_2 be the minimum and maximum slopes of the edges on the external face of Γ. We say that Γ has *tolerance angle* α if the maximum deviation of the slope of any edge of Γ from the interval $[\theta_1, \theta_2]$ is upper bounded by α; i.e.,

$$\max(\theta(e) - \theta_2, \theta_1 - \theta(e)) \leq \alpha \quad \text{for each edge } e \in E.$$

Theorem 3.1. *Let G be a planar embedding of a triangular st-graph with the edge (s, t) on the external face. Given an upward drawing Δ for the external face of G, and a positive constant α, there exists an upward drawing Γ of G with external face Δ and tolerance angle α.*

Proof. The proof is by induction on the number n of vertices of G. The basis of the induction, $n = 3$, is immediate. Now, assume that the theorem holds for graphs with less than n vertices. Let v be a vertex of G that is not on the external face, and let $v_0, \ldots, v_{k-1}$ be the neighbors of v, in circular order around v. Since G is triangulated, these vertices form a simple undirected cycle χ. We distinguish two cases:

Case 1: χ has a *chord*, i.e., an edge (v_i, v_j) between two nonconsecutive vertices. The cycle $\lambda = (v_i, v, v_j)$ delimits two subgraphs of G, where we denote the external one with G_1 and the internal one with G_2 (see Fig. 8). Observe that both G_1 and G_2 are triangular st-graphs and have less than n vertices. We use the inductive hypothesis to construct an upward drawing Γ_1 for G_1, with external face Δ and tolerance angle $\frac{1}{2}\alpha$. Let A be the triangle in Γ_1 corresponding to the cycle λ. We use again the inductive hypothesis to construct an upward drawing Γ_2 for G_2, with

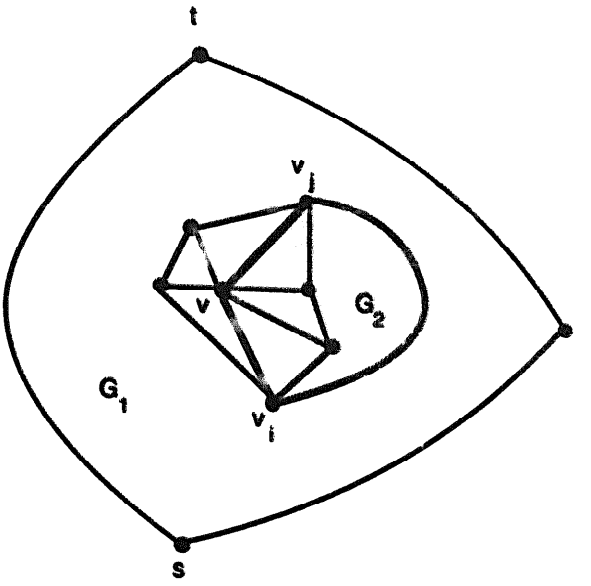

Fig. 8. Subgraphs G_1 and G_2 in the proof of Theorem 3.1.

external face Δ and tolerance angle $\frac{1}{2}\alpha$. The union of Γ_1 and Γ_2 is an upward drawing for G with the required properties.

Case 2: Otherwise. Let v_i be a predecessor of v such that there is no directed path from v_i to any other predecessor of v. Note that, given a topological ranking for the vertices of G, v_i can be chosen as the predecessor of v with highest rank. We contract edge (v_i, v) into vertex v_i, i.e., we remove vertex v together with its incident edges, and if $k \geq 4$, we add $k-3$ new edges between v_i and the other vertices of χ nonadjacent to v_i, where we add the edge (v_i, v_j) if v_j is a successor of v, and the edge (v_j, v_i) if v_j is a predecessor of v (see Fig. 9). The above choice of v_i and Lemma 2.1 ensure that the resulting graph G' is a triangular st-graph. Now, since G' has $n-1$ vertices, we apply the inductive hypothesis to construct an upward drawing Γ' for it, with external face Δ and tolerance angle $\frac{1}{2}\alpha$.

To complete the embedding, we have to remove the new edges and re-insert vertex v inside the polygon X of Γ' corresponding to the cycle χ. A legal placement for vertex v must satisfy the following requirements:

(1) every vertex of X must be properly visible from v;

(2) the slope of every edge incident upon v must be in the interval $[\theta_1 - \alpha, \theta_2 + \alpha]$.

Because of our construction, we have that polygon X and vertex v_i verify the hypotheses of Lemma 2.4. Therefore, there is a disk D centered at v_i such that all the points in the sector S defined as the intersection of D with the wedge of v_i belong to the kernel of X. Clearly, placing v in the interior of S satisfies requirement (1). It remains to be shown that v can be placed inside S so that also requirement (2) is verified.

Since the edges of G' have slopes in the interval $[\theta_1 - \frac{1}{2}\alpha, \theta_2 + \frac{1}{2}\alpha]$, the angle at vertex v_i delimited by the lines with slopes $\theta_1 - \alpha$ and $\theta_2 + \alpha$ has a nonempty intersection S' with S. In fact, if the neighbors of v_i in χ are both predecessors of

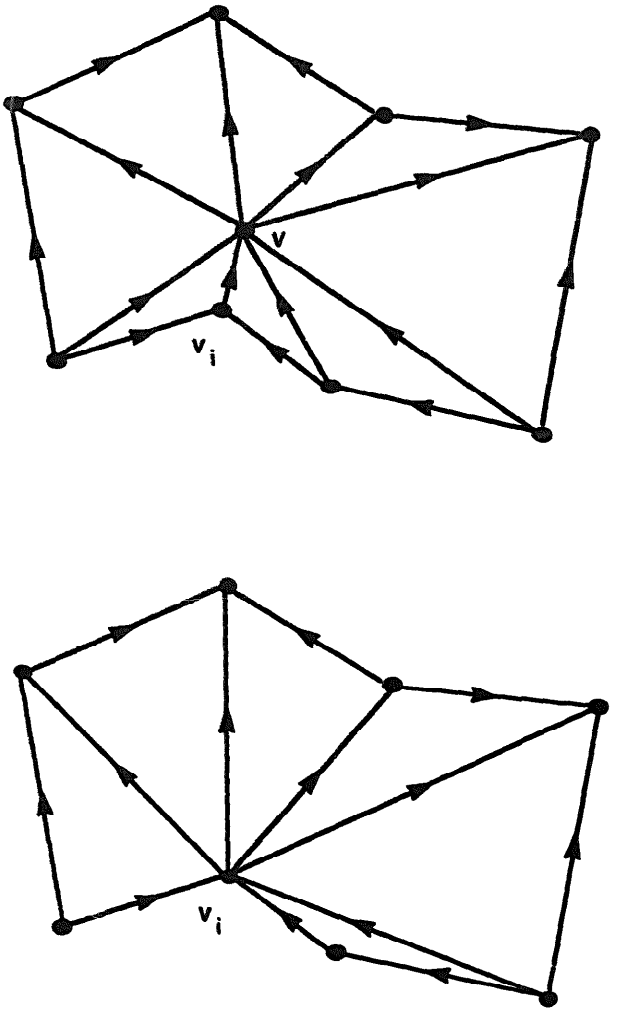

Fig. 9. Contraction of edge (v_i, v) in the proof of Theorem 3.1.

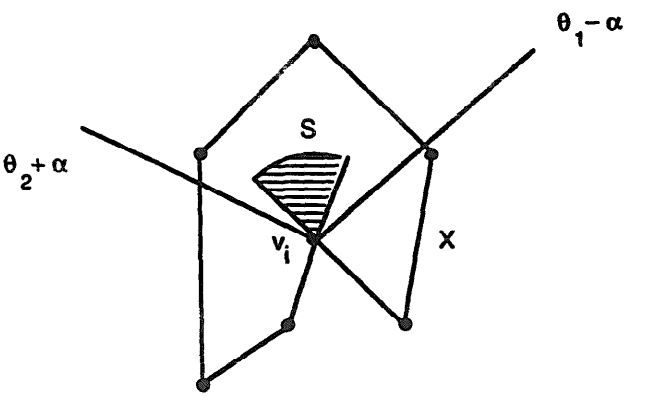

Fig. 10. Example of sector S in the proof of Theorem 3.1.

of v, we have $S' = S$. Figure 10 shows an example of this. Therefore, placing v inside S' ensures that condition (2) is verified for the edge (v_i, v). Now, for $j \neq i$, define the *mate* of the edge of G between v and v_j as the edge of G' between v_i and v_j. Since v_i belongs to the boundary of S', given any small $\varepsilon > 0$ there exists a point p in S' such that placing v at p causes the slope of any edge incident upon v to differ from the slope of its mate by at most ε. Therefore, since the inductive hypothesis ensures that the mates have slopes in the interval $[\theta_1 - \frac{1}{2}\alpha, \theta_2 + \frac{1}{2}\alpha]$, the choice $\varepsilon = \frac{1}{2}\alpha$ gives a position for v such that condition (2) is fulfilled for all the edges incident upon v. □

The algorithm for upward drawings is based on the proof of Theorem 3.1, and consists of two phases, *Preprocessing* and *Drawing*: the Preprocessing phase builds the data structures used by the algorithm; the Drawing phase is a recursive procedure that generates the actual drawing of the graph. Now, we describe the two phases in more detail.

Preprocessing. Let G be a planar st-graph. First, we find a planar embedding for G such that the edge (s, t) is on the boundary of the external face, and we compute a topological ranking for the vertices of G, e.g., a topological sorting. Then, we add further edges in order to transform G into a triangular st-graph, as follows: For each face f of G, let low(f) be the vertex on the boundary of f with lowest rank; we add directed edges from low(f) to the remaining vertices of f nonadjacent to low(f). From Lemma 2.2, these edges do not introduce cycles. Finally, we build the following data structures:

(1) A set of circular *adjacency lists* $A(v)$ that give the sequences of edges incident upon each vertex v, in counterclockwise order around v.

(2) A set of balanced search trees $T(v)$, called *neighbor trees*, that store the edges incident upon each vertex v sorted according to the rank of their other endpoint vertex. Notice that we can perform insertions, deletions, and searches in $T(v)$ in $O(\log n)$ time.

(3) A doubly connected list L, called the *candidate list*, containing the internal vertices of G with degree at most 5. Notice that L is not empty since every maximal planar graph has at least four vertices of degree 5 or less (a consequence of the fact that there are $3n - 6$ edges).

Drawing. The Drawing phase is realized with a recursive procedure that follows with some variations the steps of the proof of Theorem 3.1. We efficiently discriminate between Case 1 and Case 2 by choosing v from the candidate list L, and then searching for chords using the neighbor trees. Since v has degree at most 5, we need to perform only a constant number of such tree searches, with total time $O(\log n)$. In Case 1, in order to call the procedure recursively, we have to create the candidate lists L_1 and L_2 for the subgraphs G_1 and G_2. This is done by visiting in parallel G_1 and G_2, where at each visit step we perform a constant amount of work. Whenever we find a vertex that is the original candidate list L of G we move it to the appropriate

list L_1 or L_2. We terminate this parallel visit as soon as one of the two subgraphs is completely visited. At this point, we move to the candidate list of the other subgraph the remaining vertices in L. A detailed Pascal-like description of the Drawing phase is shown in Fig. 11. The subroutine VISITSTEP(G_i, L_i) performs one step of the visit of graph G_i as discussed above. At the end of the Drawing phase, we remove from the drawing of G the triangulation edges added in the Preprocessing step.

```
Procedure UPWARD_DRAW(G, L, Δ, α)
{constructs an upward drawing of the triangular st-graph G}
{G is represented by the adjacency lists A(v) and the neighbor trees T(v)}
{L points to the list of internal vertices of G with degree at most five}
{Δ is the assigned external face; α is the tolerance angle}
begin
  if G has three vertices
  then draw G as Δ
  else begin
    SELECT a vertex v from L;
    SCAN A(v) to find the predecessor u of v with highest number;
    TEST if the cycle χ of the neighbors of v has a chord with endpoint u by searching in the neighbor
      tree T(u);
    if such a chord (u, w) exists
    then begin {Case 1}
      let G1 and G2 be the subgraphs of G external and
      internal to the cycle λ = (u, v, w) respectively;
      DECOMPOSE the adjacency list structure of G into the structures for G1 and G2 (the vertices and
        edges of λ are duplicated);
      SPLIT the list L into the lists L1 and L2, as follows:
        repeat
          VISITSTEP(G1, L1);
          VISITSTEP(G2, L2);
        until visit of G1or G2 is completed;
        MOVE the remaining elements of L to the list Li of the graph Gi whose visit has not been
        completed;
        REMOVE, possibly, u, v, and w from L1;
      UPWARD_DRAW(G1, L1, Δ, ½α);
      let Λ be the drawing of cycle λ in G1;
      UPWARD_DRAW(G2, L2, Λ, ½α);
      MERGE the adjacency lists and candidate lists of G1 and G2;
    end {then}
    else begin {Case 2}
      CONTRACT edge (u, v), i.e.:
        remove vertex v and its incident edges;
        add new edges incident upon u to triangulate cycle χ;
        modify accordingly the neighbor trees of the vertices of χ;
        modify accordingly the list L;
      UPWARD_DRAW(G, L, Δ, ½α);
      REINTRODUCE edge (u, v) and restore the data structures of G;
      PLACE in suitable position vertex v;
    end {else}
  end {else}
end {UPWARD_DRAW}
```

Fig. 11. Algorithm for constructing an upward drawing.

Lemma 3.2. *Let G be an st-graph, ρ a topological ranking for G, and (u, v) an edge of G such that u is a predecessor of v with highest ρ value. If G' is the graph obtained from G by contracting the edge (u, v) into vertex u, then the restriction of ρ to G' is a topological ranking for G'.*

Proof. The contraction of edge (u, v) into u removes the edges incident upon v, and introduces new edges incident upon u and the former neighbors of v. From the definition of u, if the new edge (w, u) is ingoing into u, then $\rho(w) < \rho(u)$. Conversely, if the new edge (u, w) is outgoing from u, then w is a former successor of v, and hence, $\rho(v) < \rho(w)$, which implies also $\rho(u) < \rho(w)$. □

Theorem 3.3. *Let G be a planar st-graph with n vertices. The above algorithm constructs an upward drawing for G in $O(n \log n)$ time and $O(n)$ space.*

Proof. The correctness of the algorithm derives from Lemma 3.2 and the proof of Theorem 3.1. With regard to the time complexity, we consider first the Preprocessing phase: A planar representation for G can be constructed in $O(n)$ time using the algorithm described in [5]. The triangulation of G and the subsequent topological sorting can be performed in $O(n)$ time since a triangular st-graph has $3n-6$ edges. Finally, the use of a bucket-sort technique to sort by rank the neighbors of each vertex allows us to construct the neighbor trees in $O(n)$ time. Therefore, the Preprocessing phase can be completed in $O(n)$ time.

Now, we turn our attention to the Drawing phase. The correctness follows from Lemmas 2.1, 2.2, and 2.4. The following analysis makes critical use of the fact that vertex v has bounded degree. Clearly, step SELECT takes constant time. Step SCAN has to consider at most five edges, so that it takes constant time too. Step TEST consists of performing at most three searches in the neighbor tree $T(u)$, with total time $O(\log n)$.

In Case 1, step DECOMPOSE can be performed in constant time since G_1 and G_2 are separated by a cycle of three vertices. For step SPLIT, the complexity is $O(\min(n_1, n_2))$, where n_1 and n_2 are the number of vertices of G_1 and G_2 respectively. Notice that substep MOVE can be implemented in constant time by appending L to L_i. Finally, MERGE takes constant time.

In Case 2, step CONTRACT performs constant time work to modify the adjacency list and update L, and at most ten deletions and six insertions in the neighbor trees of v and the vertices of cycle χ. This takes $O(\log n)$ time. Step REINTRODUCE performs the inverse operations of step CONTRACT, and thus takes $O(\log n)$ time too. Finally, step PLACE requires the computation of the position v inside a polygon with at most five vertices, and can be performed in constant time.

From the above considerations, the time complexity of the Drawing phase is expressed by the following recursive formula, where $N = n-3$ denotes the number of internal vertices of G:

$$T(N) = O(\log N) + \max_{0 \le K \le \frac{1}{2}N} \{T(K) + T(N-K) + O(K)\}$$

whose solution is $T(N) = O(N \log N)$. This completes the proof of the $O(n \log n)$ time bound.

For the space occupation, we observe that the data structures require $O(n)$ space, and that the number of recursive calls is at most n, so that also the depth of the recursion stack is $O(n)$. □

Theorem 3.4. *Let G be a planar st-graph with n vertices, m edges, and r faces. There is an algorithm that constructs in $O(n)$ time a directed visibility representation Γ for G such that all segment endpoints are grid points,* $\textsc{Height}(\Gamma) \leq n-1$, *and* $\textsc{Width}(\Gamma) \leq r-1 \leq 2n-5$.

Proof. The algorithm, shown in Fig. 12, is a straightforward modification of the one for constructing visibility representations of undirected planar graphs given in [25]. The correctness follows from Lemmas 2.1–2.3. Using the *critical path method* [12], step (3) can be performed in $O(n)$ time. Clearly, also the other steps take linear time. Therefore, the overall complexity of the algorithm is $O(n)$.

The height and width of the directed visibility representation constructed are given by $\alpha(t)$ and $\beta(t^*)$. Now, from the definition of these functions, we have that

$$\alpha(t) \leq n-1, \qquad \beta(t^*) \leq r-1 \leq 2n-5.$$

This completes the proof of the theorem. □

An example of the construction performed by Algorithm VISIBILITY_DRAW is shown in Fig. 13. A directed visibility representation constructed by the above algorithm can be modified to obtain a monotonic grid drawing, using the algorithm shown in Fig. 14. A possible choice for the function $P(v)$ is:

$$P(v) = (\lfloor \tfrac{1}{2}(x_L + x_R) \rfloor, Y(v))$$

Algorithm VISIBILITY_DRAW
begin
(1) Find a planar representation $\hat{G}$ of G such that the arc (s, t) is on the external face, and the rest of G lies on the right side of (s, t).
(2) Construct the dual graph G^* of $\hat{G}$. The source and sink of G^* are denoted with s^* and t^* respectively.
(3) Compute, for every vertex v of G, the length $\alpha(v)$, of a longest path in G from s to v. Similarly, compute, for every vertex f of G^*, the length $\beta(f)$ of a longest path in G^* from s^* to f.
(4) Let RIGHT(u, v) be the face on the right of edge (u, v).
for each edge (u, v) **do**
draw a vertical segment with abscissa $X(u, v) = \beta(\text{RIGHT}(u, v))$ between ordinates $\alpha(u)$ and $\alpha(v)$;
(5) **for each** vertex v **do**
draw a horizontal segment with ordinate $Y(v) = \alpha(v)$ between the leftmost and rightmost abscissas of the vertical segments associated with the edges incident upon v.
end

Fig. 12. Algorithm for constructing a directed visibility representation.

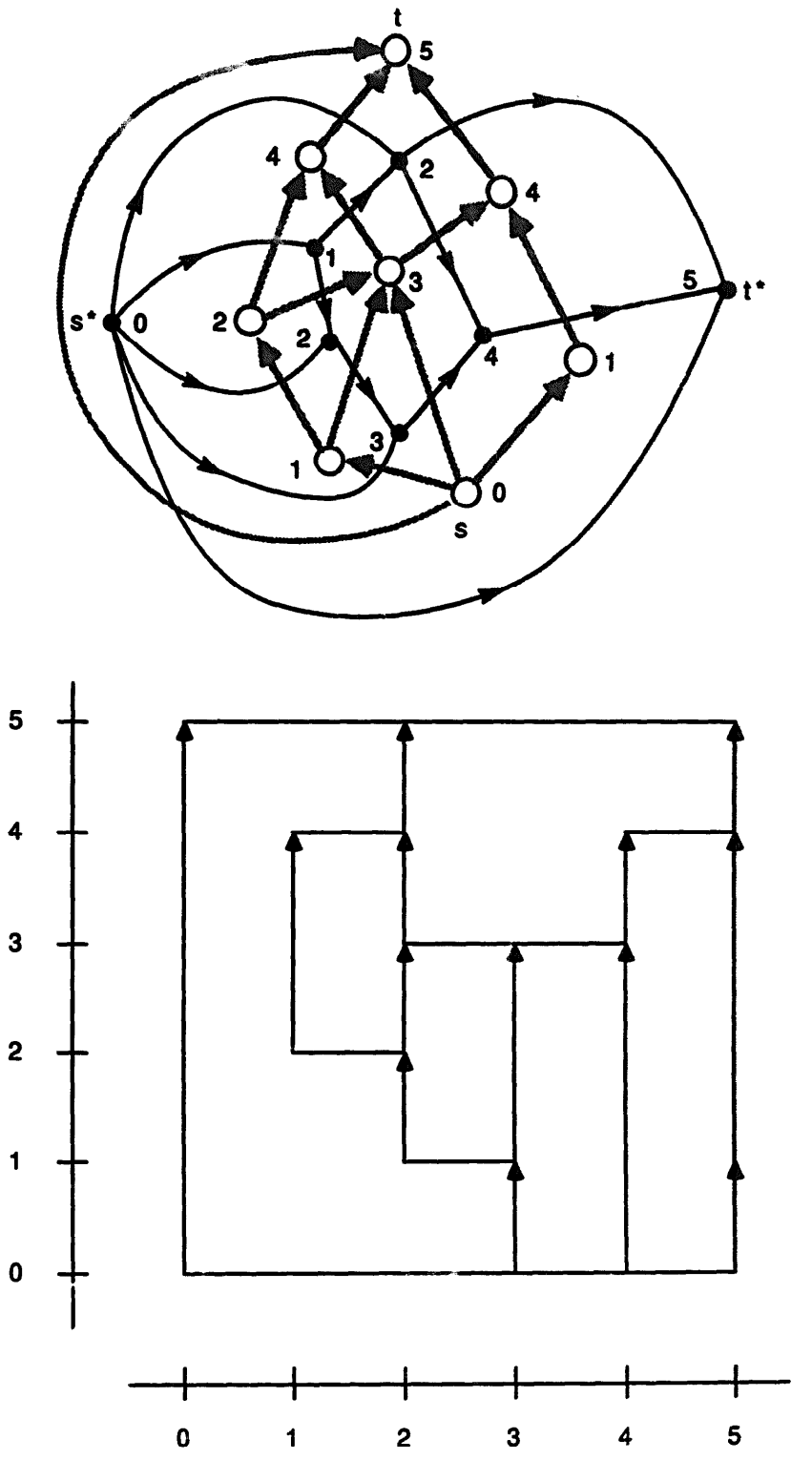

Fig. 13. Example of construction of a directed visibility representation performed by algorithm VISIBILITY_DRAW.

Algorithm GRID_DRAW
begin
(1) Construct a directed visibility representation of G using Algorithm VISIBILITY_DRAW;
we denote with $Y(v)$ the ordinate of vertex-segment $\sigma(v)$, and with $X(u, v)$ the abscissa of edge-segment $\sigma(u, v)$.
(2) **for each** vertex v **do**
replace the vertex-segment $\sigma(v)$ with a point $P(v)$ on $\sigma(v)$;
endfor
(3) **for each** edge (u, v) **do**
if $Y(v) - Y(u) = 1$ {*short* edge}
then
replace the edge-segment $\sigma(u, v)$ with the segment $P(u) \rightarrow P(v)$;
else begin {*long* edge}
replace the edge-segment $\sigma(u, v)$ with the polygonal line:
$P(u) \rightarrow (X(u, v), Y(u)+1) \rightarrow (X(u, v), Y(v)-1) \rightarrow P(v)$;
end
endfor
end

Fig. 14. Algorithm for constructing a monotonic grid drawing.

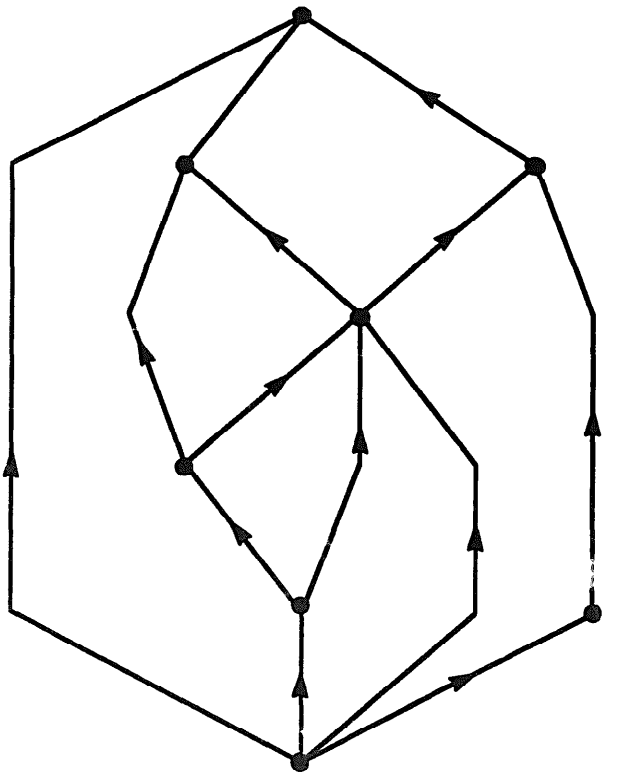

Fig. 15. Example of monotonic grid drawing produced by algorithm GRID_DRAW from the directed visibility representation of Fig. 13 using the median positioning.

where x_L and x_R are the abscissas of the left and right endpoint of segment $\sigma(v)$ respectively. An example of monotonic grid drawing obtained with this choice from the directed visibility representation of Fig. 13 is shown in Fig. 15.

Theorem 3.5. *Let G be a planar st-graph with $n \geq 3$ vertices, m edges, and r regions. For any positioning function $P(v)$, Algorithm* GRID_DRAW *constructs in $O(n)$ time a monotonic grid drawing Γ for G such that* HEIGHT$(\Gamma) \leq n-1$, WIDTH$(\Gamma) \leq r-1 \leq 2n-5$, *and* BENDS$(\Gamma) \leq 2r-2 \leq 4n-10$. *Also, every edge has at most two bends.*

Proof. The correctness of the algorithm can be proved with simple geometric considerations. The bounds on the height and width follow from the corresponding properties of the directed visibility representation produced by Algorithm VISIBILITY_DRAW. Each edge-segment $\sigma(u, v)$ of the directed visibility representation is replaced by either a segment or a polygonal line with at most two bends. The first case occurs when the edge is *short*, i.e., $Y(v) - Y(u) = 1$. From steps (3)-(5) of Algorithm VISIBILITY_DRAW, $Y(v)$ is equal to the length of a longest path from s to v. Hence, for each vertex v distinct from s, there is a short edge (u, v) ingoing into v. This implies that there are at least $n-1$ short edges. Therefore, the total number of bends is at most:

$$\text{BENDS}(\Gamma) \leq 2(m-n+1) = 2(r-1) \leq 4n-10. \qquad \square$$

The above bound on the number of bends can be improved using a different choice for the function $P(v)$, as shown by the following theorem.

Theorem 3.6. *Let G be a planar st-graph with $n \geq 4$ vertices, m edges, and r regions. Also, let $m_l \geq 3$ be the number of long edges in the directed visibility representation of G. If $P(v)$ is chosen as the intersection of vertex-segment $\sigma(v)$ with the edge-segment of a long edge incident upon v, whenever one exists, then Algorithm* GRID_DRAW *produces a monotonic grid drawing Γ with*

$$\text{BENDS}(\Gamma) \leq \tfrac{5}{3}m_l - 2 \leq \tfrac{1}{3}(5r-11) \leq \tfrac{1}{3}(10n-31).$$

Proof. The above choice eliminates at least one bend for each vertex v that is incident upon a long edge. Since the long edges form a planar graph, the number n_l of such vertices is at least $n_l \geq \frac{1}{3}(m_l+6)$. Hence, the total number of bends is $\text{BENDS}(\Gamma) \leq 2m_l - n_l \leq \frac{5}{3}m_l - 22$. The proof is completed recalling that $m_l \leq r-1$. $\square$

We show in Fig. 16 the monotonic grid drawing produced by Algorithm GRID_DRAW from the directed visibility representation of Fig. 13 using the improved positioning of Theorem 3.6.

As a direct consequence of Theorem 3.5, we have a substantial improvement over Woods' algorithm [35] for constructing grid drawings of undirected planar graphs as shown in the following corollary.

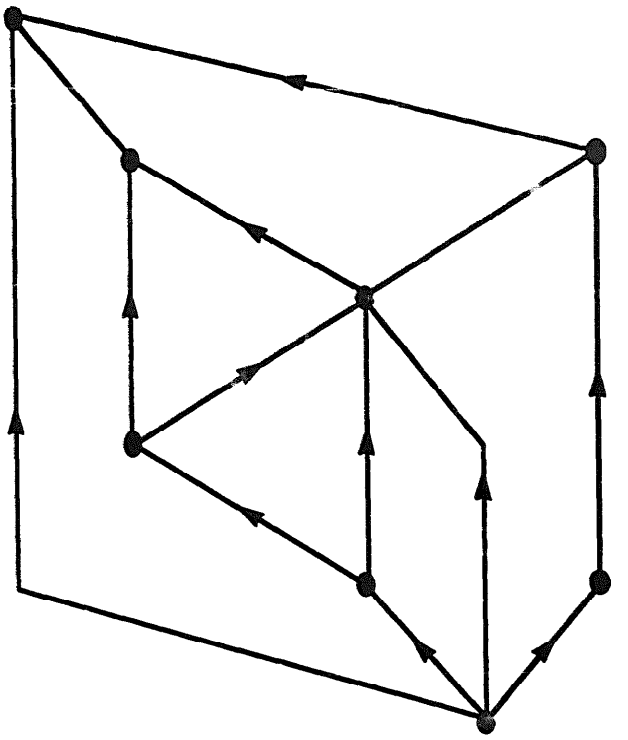

Fig. 16. Example of monotonic grid drawing produced by Algorithm GRID_DRAW from the directed visibility representation of Fig. 13 using the improved positioning of Theorem 3.6.

Corollary 3.7. *Let G be a planar undirected graph with $n \geq 3$ vertices. There is an algorithm that constructs in $O(n)$ time a grid drawing for G with height, width and number of bends bounded by $n-1$, $2n-5$, and $\frac{1}{3}(10n-31)$ respectively. Also, every edge has at most two bends.*

Proof. First, add edges to G to make it biconnected. Then orient its edges so that the resulting graph is an st-graph. This can be done in $O(n)$ time computing an st-numbering for G [11]. Finally, use Algorithm GRID_DRAW modified in such a way that it does not draw the edges initially added to G. □

4. A characterization of monotonic drawings

Planarity and acyclicity are not a sufficient condition for the existence of a monotonic drawing: two examples of planar acyclic digraphs that do not admit a monotonic drawing are shown in Fig. 17. In this section we present a characterization of the class of digraphs that admit a monotonic drawing. Namely, we show that this class consists exactly of the subgraphs of planar st-graphs, and that every graph in this class also admits a monotonic grid drawing, an upward drawing, and a directed visiblity representation.

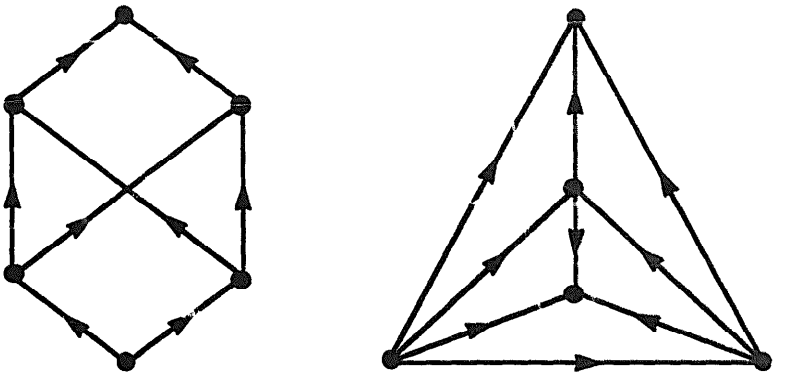

Fig. 17. Examples of planar acyclic digraphs that do not admit a monotonic drawing.

Lemma 4.1. *If G admits a monotonic drawing, then G is a subgraph of a planar st-graph.*

Proof. Let D be a monotonic drawing for G. Clearly, G is planar and acyclic. We show that we can extend D to a monotonic drawing of a planar st-graph. We denote the vertices of D with $v_1, \ldots, v_n$, sorted from bottom to top, and with $Y_1, \ldots, Y_n$ their ordinates. Let v_i be a sink vertex of D with ordinate distinct from Y_1 and Y_n, and consider a horizontal strip of the plane delimited by the horizontal lines at ordinates Y_i and $Y_i + \varepsilon$, with $\varepsilon > 0$ small enough so that the strip contains no vertices in its interior. This strip is subdivided by the edges of D into connected regions bounded by vertically monotonic curves. Now, let (v_k, v_l) be an edge on the boundary of the region that contains v_i. We add to G the edge (v_i, v_l), and we draw it onto D as a monotonic curve close to the edge (v_k, v_l) (see Fig. 18). By repeating this procedure, we can eliminate all the sinks, except the ones at ordinate Y_n. Similarly, we can eliminate all the sources, except the ones at ordinate Y_1. To complete the construction, we add two new vertices, s and t, at ordinates $Y_1 - 1$ and $Y_n + 1$

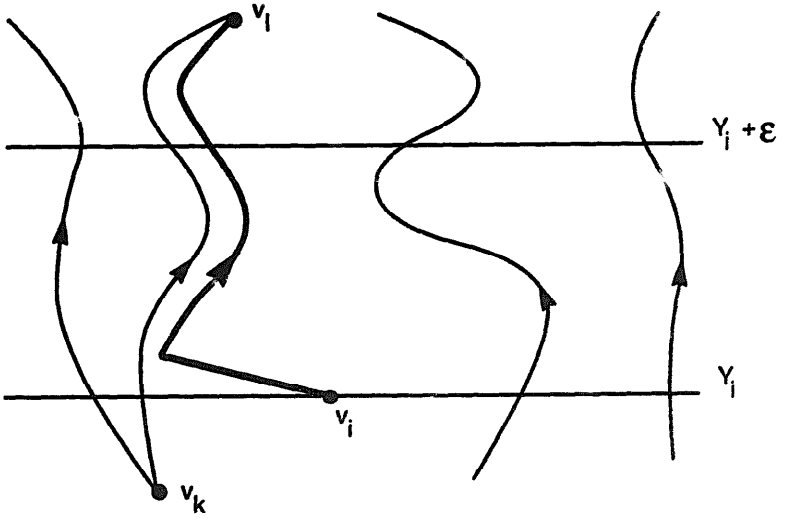

Fig. 18. Drawing the edge (v_i, v_l) in the proof of Lemma 4.1.

respectively, and we connect s to all the remaining sources and t to all the remaining sinks. □

Lemma 4.2. *If G admits a directed visibility representation, then G is a subgraph of a planar st-graph.*

Proof. Let Γ be a directed visibility representation for G. We draw two horizontal segments, σ and τ, such that Γ is contained in the interior of the rectangle whose bottom and top sides are σ and τ respectively. Now, consider the directed visibility representation Γ' defined as follows:

(1) the vertex-segments of Γ' are the vertex-segments of Γ, plus the segments σ and τ;

(2) there is an edge-segment from vertex-segment α to vertex-segment β if and only if α and β can be joined by a vertical segment that has its lower endpoint on α, has its higher endpoint on β, and does not intersect any other vertex-segment.

Γ' can be considered as the vertical visibility graph of its horizontal segments, and represents a digraph G' that clearly contains G as a subgraph. To complete the proof, we show that G' is a planar st-graph. Clearly, since edges are oriented from bottom to top, G' is acyclic. Now, observe that any vertical line drawn onto Γ' corresponds to a directed path in G' from vertex s represented by segment σ, to the vertex t represented by segment τ. Thus, every vertex of G' lies on a directed path from s to t. This implies that the only source of G' is s, and the only sink is t. Finally, the planarity of G' can be easily derived from the planar structure of Γ'. □

By combining the results of the previous section with the above two lemmas, we obtain the aforementioned characterization.

Theorem 4.3. *Let G be a digraph. The following statements are equivalent:*

(1) *G is a subgraph of a planar st-graph;*
(2) *G admits a monotonic drawing;*
(3) *G admits a monotonic grid drawing;*
(4) *G admits an upward drawing;*
(5) *G admits a directed visibility representation.*

5. Drawing algorithms for lattices

This section deals with covering digraphs of planar lattices. Let G be such a digraph; first, we give an algorithm that constructs in $O(n)$ time an upward drawing of G; then, we show a new version of Algorithm GRID_DRAW that constructs a monotonic grid drawing of G with at most $n-3$ bends. These results improve upon the corresponding ones for general st-graphs given in Section 3.

The algorithm for constructing an upward drawing is shown in Fig. 19. The drawing is generated by adding one face at a time, according to the partial order

Algorithm LATTICE_UPWARD_DRAW
begin
(1) Add to G the arc (s, t), and find a planar representation $\hat{G}$ of G such that the arc (s, t) is on the external face, and the rest of $\hat{G}$ lies on the right side of (s, t).
(2) Construct the dual graph G^* of $\hat{G}$, denoting by s^* and t^* the faces to the right and left of (s, t) respectively.
(3) Compute, for every vertex v of G, the length $\alpha(v)$ of a longest path in G from s to v. Similarly, compute for every vertex f of G^*, the length $\beta(f)$ of a longest path in G^* from s^* to f.
(4) Sort the vertices of G^* by increasing values of β, and denote them with $s^* = f_1, f_2, \ldots, f_r = t^*$.
(5) Set, for each vertex v, $Y(v) = \alpha(v)$.
    foreach vertex v of *rpath* (f_1) **do**
      $X(v) = 0$;
(6) **for** $i = 2$ to $r-1$ **do**
    begin {draw the right path of face f_i}
      compute the maximum abscissa x_a of the vertices of *lpath*(f_i);
      let *rpath*$(f_i) = (v_1, v_2, \ldots, v_{q-1}, v_q)$; {notice that $Y(v_1) \leq \cdots \leq Y(v_q)$}
      compute the abscissa x_b of the intersection between the line $y = Y(v_2)$ and the right supporting line of *lpath*(f_i) through vertex v_1;
      compute the abscissa x_c of the intersection between the line $y = Y(v_{q-1})$ and the supporting line of *lpath*(f_i) through vertex v_q;
      foreach vertex v of *rpath*(f_i) **do**
        $X(v) = \max(x_a, x_b, x_c) + 1$
    end; {for}
end

Fig. 19. Algorithm for upward drawings of planar lattices.

given by the dual graph. For the terminology used in the algorithm, recall from Lemma 2.2 that each face f of an st-graph consists of two directed paths. We will denote the *left path* and *right path* with *lpath*(f) and *rpath*(f) respectively. Also, let L be a polygonal line monotonically increasing in the vertical direction, and p an extreme point of L. We define the *right supporting line* of L through p as the straight line l such that

(1) l contains p;
(2) the open half-plane to the right of l does not contain any point of L; and
(3) l contains a point of L distinct from p.

Theorem 5.1. *Let G be the covering digraph of a planar lattice with n vertices. Algorithm* LATTICE_UPWARD_DRAW *constructs in* $O(n)$ *time and space an upward drawing of G.*

Proof. The correctness of the algorithm is based on the following facts:

(1) The right path of every face has length at least 2; this follows from the absence of transitive edges in G.

(2) Before the jth iteration of the **for** loop of step (6), the left path *lpath*(f_j) of face f_j has already been drawn. In fact, each arc of *lpath*(f_j) belongs also to the right path *rpath*(f_i) of some other face f_i. Because of the construction of the dual D^*, and the sorting of the faces in step (4), we have that $i < j$. Hence, *rpath*(f_i) has been already drawn.

Regarding the time complexity, steps (1)-(5) clearly take $O(n)$ time. For step (6), the running time is proportional to $\sum_{i=2}^{r-1}(|lpath(f_i)|+|rpath(f_i)|) \leq 2m$. Therefore, the overall time complexity is O(n). □

Theorem 5.2. *Let G be the covering digraph of a planar lattice with $n \geq 3$ vertices and m edges. Suppose that in the algorithm* GRID_DRAW *the positioning function $P(v)$ is chosen as the right endpoint of vertex-segment $\sigma(v)$. Then the resulting monotonic grid drawing Γ is such that* $BENDS(\Gamma) \leq m-n+1 \leq n-3$.

Proof. First, we show that with this choice every edge has at most one bend. Consider a face f and its left path $lpath(f)$. Since G does not have transitive edges, $lpath(f)$ consists of at least two edges. In the directed visibility representation constructed in the first step of the algorithm, the edge-segments of $lpath(f)$ lie on the same vertical line (see Fig. 13). Because of the choice of $P(v)$, all the vertices of this path, except the first and the last, lie on this line. Hence, the first and last edge of $lpath(f)$ have at most one bend, and the other edges do not have bends.

The nontransitivity of G implies that G has at most $2n-4$ edges. We thus have $BENDS(\Gamma) \leq m \leq 2n-4$. Recalling from the proof of Theorem 3.5 that there are at least $n-1$ short edges that are always drawn without bends, we obtain $BENDS(\Gamma) \leq m-n+1 \leq n-3$. □

6. Conclusions and open problems

We have investigated several plane representations of acyclic digraphs such that edges flow in the same direction. The results presented in this paper can be summarized as follows:

(1) We have presented a complete characterization of the class of planar digraphs that admit a monotonic drawing.

(2) We have shown that this class is the same as the one of planar digraphs that admit a monotonic grid drawing, an upward drawing, or a directed visibility representation.

(3) We have given efficient algorithms for constructing these representations, whose complexity is $O(n \log n)$ for upward drawings, and $O(n)$ for monotonic grid drawings and directed visibility representations.

(4) Finally, for covering digraphs of lattices, we have given an $O(n)$ time algorithm for the construction of upward drawings.

The algorithms for monotonic grid grid drawings have the additional feature that the drawing produced has $O(n^2)$ area and $O(n)$ bends, and are therefore especially attractive for practical applications.

Some open problems still remain in this area. First, is there a linear-time algorithm for constructing upward drawings? Another important question raised by our result is to find an efficient algorithm to test if a digraph G is a subgraph of some st-graph. Finally, the algorithm for constructing upward drawings might require exponential precision in the numerical computations of the coordinates; i.e., if we want the vertices to be placed at grid points, we might have a drawing with exponential area. This drawback is common to all existing algorithms that draw planar graphs with straight lines [4, 6, 31, 34], and it is an outstanding open problem whether any n-vertex planar graph admits a straight-line drawing with vertices placed at grid points and area bounded by a polynomial in n.

A problem closely related to the one discussed in this paper, and arising in many applications, regards straight-line drawings of acyclic digraphs with vertices arranged into levels (parallel lines), so that the successors of every vertex v are placed at levels above the level of v [21, 24, 33]. Given the assignment of the vertices to the levels, it is interesting to test if the graph admits a planar straight-line drawing subject to the above constraint on the vertex positions. A first step in this direction can be found in [7], where a linear time planarity testing algorithm is presented for the case of digraphs with exactly one source, and with edges connecting vertices across consecutive levels. In the case of non-planarity, it is meaningful to attempt to minimize the number of crossings. Eades et al. have shown that this problem is NP-hard even when the levels are two [8], and have provided several crossing minimization heuristics [9, 10]. Other heuristics for crossing minimization are described in [3, 21, 24, 33]. It would be interesting to devise approximation guarantee algorithms for crossing minimization.

Acknowledgment

We are indebted to Enrico Nardelli for many helpful discussions. We are also grateful to Herbert Edelsbrunner and Ioannis Tollis for their comments. Special thanks to Carlo Batini for his continuous encouragement during this research work.

References

[1] C. Batini, E. Nardelli and R. Tamassia, A layout algorithm for data-flow diagrams, *IEEE Trans. Software Engineering* SE-12 (1986) 538–546.

[2] G. Birkhoff, *Lattice Theory*, American Mathematical Society Colloquium Publications 25 (Amer. Mathematical Soc., Providence, RI, 1967).

[3] M. Carpano, Automatic display of hierarchized graphs for computer aided decision analysis, *IEEE Trans. Systems, Man, and Cybernetics* SMC-10 (1980) 705–715.

[4] N. Chiba, T. Yamanouchi and T. Nishizeki, Linear algorithms for convex drawings of planar graphs, in: J.A. Bondy and U.S.R. Murty, eds., *Progress in Graph Theory* (Academic Press, New York, 1984) 153–173.

[5] N. Chiba, T. Nishizeki, S. Abe and T. Ozawa, A linear algorithm for embedding planar graphs using PQ-trees, *J. Comput. System Sci.* 30 (1985) 54–76.

[6] N. Chiba, K. Onoguchi and T. Nishizeki, Drawing planar graphs nicely, *Acta Inform.* **22** (1985) 187–201.
[7] G. Di Battista and E. Nardelli, An algorithm for testing planarity of hierarchical graphs, in: G. Tinhofer and G. Schmidt, eds., *Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science **246** (Springer, Berlin, 1987) 277–289.
[8] P. Eades, B. McKay and N. Wormald, An NP-hard crossing number problem for bipartite graphs, Tech. Rept. 60, Dept. of Computer Science, Univ. of Queensland, 1985.
[9] P. Eades and D. Kelly, Heuristics for reducing crossings in 2-layered networks, *Ars Combin.* **21** (1986) 89–98.
[10] P. Eades and N. Wormald, The median heuristics for drawing 2-layered networks, Tech. Rept. 69, Dept. of Computer Science, Univ. of Queensland, 1986.
[11] S. Even and R.E. Tarjan, Computing an *st*-numbering, *Theoret. Comput. Sci.* **2** (1976) 339–344.
[12] S. Even, *Graph Algorithms* (Computer Science Press, Rockville, MD, 1979).
[13] I. Fary, On straight lines representations of planar graphs, *Acta Sci. Math. Szeged* **11** (1948) 229–233.
[14] B. Grunbaum and G. Shephard, The geometry of planar graphs, in: *Proc. 8th British Combinatorial Conf.* (1981) 124–150.
[15] D. Kelly and I. Rival, Planar lattices, *Canad. J. Math.* **27** (1975) 636–665.
[16] D. Kelly, On the dimension of partially ordered sets, *Discrete Math.* **35** (1981) 135–156.
[17] D. Kelly, Fundamentals of planar ordered sets, *Discrete Math.* **63** (1987) 197–216.
[18] R.H.J.M. Otten and J.G. van Wijk, Graph representations in interactive layout design, in: *Proc. IEEE Internat. Symp. on Circuits and Systems*, New York (1978) 914–918.
[19] C. Platt, Planar lattices and planar graphs, *J. Combin. Theory Ser. B* **21** (1976) 30–39.
[20] P. Rosenstiehl and R.E. Tarjan, Rectilinear planar layouts of planar graphs and bipolar orientations, *Discrete & Comput. Geom.* **1** (1986) 342–351.
[21] L. Rowe, M. Davis, E. Messinger, C. Meyer, C. Spirakis and A. Tuan, A browser for directed graphs, Manuscript, EECS Dept., Univ. of California, Berkeley, 1986.
[22] S.K. Stein, Convex maps, *Proc. Amer. Math. Soc.* **2** (1951) 464–466.
[23] J.A. Storer, On minimal node-cost planar embeddings, *Networks* **14** (1984) 181–212.
[24] K. Sugiyama, S. Tagawa and M. Toda, Methods for visual understanding of hierarchical systems, *IEEE Trans. on Systems, Man, and Cybernetics* **SMC-11** (1981) 109–125.
[25] R. Tamassia and I.G. Tollis, A unified approach to visibility representations of planar graphs, *Discrete & Comput. Geom.* **1** (1986) 321–341.
[26] R. Tamassia, On embedding a graph in the grid with the minimum number of bends, *SIAM J. Comput.* **16** (1987) 421–444.
[27] C. Thomassen, Planarity and duality of finite and infinite planar graphs, *J. Combin. Theory Ser. B* **29** (1980) 244–271.
[28] C. Thomassen, Plane representations of graphs, in: J.A. Bondy and U.S.R. Murty, eds., *Progress in Graph Theory* (Academic Press, New York, 1984) 43–69.
[29] W. Trotter and J. Moore, Jr., The dimension of planar posets, *J. Combin. Theory Ser. B* **22** (1977) 54–67.
[30] W.T. Tutte, Convex representations of graphs, *Proc. London Math. Soc.* **10** (1960) 304–320.
[31] W.T. Tutte, How to draw a graph, *Proc. London Math. Soc.* **3** (1963) 743–768.
[32] K. Wagner, Bemerkungen zum Vierfarbenproblem, *Jber. Deutsch. Math.-Verein* **46** (1936) 26–32.
[33] J. Warfield, Crossing theory and hierarchy mapping, *IEEE Trans. on Systems, Man, and Cybernetics* **SMC-7** (1977) 502–523.
[34] L. Woo, An algorithm for straight-line representation of planar graphs, *J. Franklin Inst.* **287** (1969) 197–208.
[35] D. Woods, Drawing planar graphs, Ph.D. Thesis (Tech. Rept. STAN-CS-82-943), Computer Science Dept., Stanford Univ., 1982.
[36] I. Yaglom and V. Boltyanskii, *Convex Figures* (Holt, Rinehart & Winston, New York, 1961).